Rakoshare, un outil de synchronisation de dossiers pour tout le monde

49

Posté par **rakoo** (**page perso**) le 21/06/14 à 12:02. Édité par **Nÿco**, **tankey**, **Xavier Claude**, **Benoît Sibaud** et **palm123**. Modéré par **Xavier Claude**.

Rakoshare est un logiciel, en Go, sous licence MIT, de synchronisation de dossiers entre plusieurs machines. Il se veut simple d'installation et d'utilisation par le plus grand nombre. Fonctionnellement, il est très largement basé sur Bittorrent Sync, un logiciel équivalent mais non-libre.

- Page du projet rakoshare sur GitHub (420 clics)
- Bittorrent Sync (76 clics)

Sommaire

- Installation
- Utilisation
- Les dessous
- Limitations
- Pourquoi ne pas avoir choisi…
  - rsync ?
  - unison ?
  - owncloud/seafile ?
  - camlistore ?
  - clearskies ou syncthing ?
  - Les autres ?
- L'avenir

Rakoshare est un projet sur lequel je travaille depuis l'annonce de Bittorrent Sync, et après avoir vu que depuis tout ce temps il n'y a même pas une annonce de l'ouverture de son code source.

Le but de cette application est de rester extrêmement simple, tant à l'installation qu'à l'utilisation, parce que ce sont là des critères essentiels à l'adoption d'un logiciel. L'une des conséquences est ainsi qu'il n'y a pas besoin d'installer un serveur central; n'importe quelle machine compatible peut se joindre au partage.

Installation

Le binaire en question est prévu, mais pour le moment il vous faudra le compiler par vous-même. Les instructions sont sur la page du projet, il s'agit tout simplement d'une application Go standard.

Utilisation

L'utilisation se résume à choisir un dossier que l'on souhaite partager, créer un *id* aléatoire (pour être sûr qu'il ne soit pas déjà choisi), et lancer l'application en liant les deux:

```
$ rakoshare -fileDir ~/myDir -id <myId> -useLPD=true -useDHT=true
```

La même commande pourra être lancée sur chaque ordinateur qui doit recevoir une copie du dossier. Notez que l'emplacement du dossier ne doit pas nécessairement être le même sur toutes les machines.

Les dessous

Comme les plus attentifs auront pu le remarquer, Rakoshare ré-utilise le protocole bittorrent qui se trouve être un excellent protocole de transport de données. Rakoshare est principalement une glue par dessus celui-ci pour permettre un plus grand dynamisme, notamment sur la manière d'informer les pairs d'un changement dans le dossier et propager ce changement. L'intérêt principal est que l'identifiant est la seule information qui a besoin d'être partagée avec les autres participants: une suite de 40 caractères est suffisante pour mettre d'accord un nombre potentiellement infini de machines

Rakoshare s'inspire également de Couchdb, qui a des idées réellement intéressantes pour la gestion des versions que je réutiliserai. D'autres technologies, comme du *multiplexing* de connexions, du chiffrement et une gestion des autorisations sont prévues pour les versions à venir.

Rakoshare n'est pas parti de zéro, je suis parti de Taipei-Torrent, un client bittorrent écrit en Go. Le choix de ce langage ne s'est d'ailleurs pas totalement fait au hasard : bien sûr, j'ai commencé à m'y intéresser quand on a commencé à parler de lui, et mes premières impressions ont été celles que tout le monde a à peu près eues: une sémantique extrêmement simple (si vous savez déjà programmer, vous n'apprendrez rien de spécialement neuf) mais suffisamment puissantes pour construire des applications performantes et fiables très rapidement. En plus de ça, les binaires créés

sont *all-inclusive*, c'est à dire qu'ils contiennent absolument tout ce qui est nécessaire; il n'y a plus de problèmes de déploiements. Enfin, la gestion de la concurrence en fait un langage tout à fait adapté pour tout logiciel qui reçoit des informations de l'extérieur à des moments indéfinis, comme par exemple un serveur.

Limitations

Rakoshare en est à sa version Make It Work: ça marche pour un dossier seulement (impossible de synchroniser plusieurs dossiers en même temps), ça passe en clair, il y a quelques instabilités, c'est verbeux et c'est en ligne de commande, il y aura certainement des problèmes si vous êtes derrière un NAT/pare-feu… mais ça marche. Je pense que ce logiciel peut servir à beaucoup de monde, donc j'ai décidé de l'annoncer pour attendre les retours des moules les plus aventureuses.

Rakoshare se limitera fortement à la synchronisation de dossiers. J'estime que ce cas d'utilisation est suffisant pour remplir la majorité des besoins (puisque tout est potentiellement fichier, et à peu près n'importe quel utilisateur comprend les principes de base du système de fichier).

Pourquoi ne pas avoir choisi…

rsync ?

rsync est le vénérable outil de transfert de dossiers, initialement connu pour son efficacité dans le protocole de transport lui-même (rsync considère à juste titre que les échanges réseau sont la partie la plus couteuse d'un échange de données, et se débrouille pour minimiser la quantité d'information à faire transiter) mais qui a su s'imposer comme le standard du transfert ad-hoc par son incroyable quantité d'options.

Malheureusement, rsync est un outil assez « bas-niveau », dans la mesure où son invocation reste relativement manuelle, et il n'est fait que pour synchroniser 2 machines entre elles.

unison ?

Unison est écrit en OCaml, un langage qui m'intéressait également, en tout cas théoriquement. Par rapport à rsync, il a l'avantage de gérer les synchronisations dans les deux sens ; malheureusement il est également fait pour les échanges entre 2 machines.

owncloud/seafile ?

L'un des prérequis que je voulais était de ne pas avoir à installer de serveur, pour garder une simplicité maximale. Mon but est bien que les gens qui utilisent Bittorrent Sync (voire Dropbox) utilisent Rakoshare; il ne faut pas passer par une étape aussi limitante.

camlistore ?

Camlistore est un projet de « dépôt » personnel d'un peu tout et n'importe quoi; lâchez-y vos photos et votre musique, et un système d'indexation vous permettra de vous y retrouver plus facilement dans votre masse d'information. Je pense que Camlistore est réellement intéressant pour ceux qui produisent/reçoivent beaucoup d'informations et souhaitent en garder une trace dans un lieu sûr; malheureusement la procédure d'installation n'est pas encore aussi simple, et dans l'état, vous devrez dupliquer vos informations entre votre système de fichiers et Camlistore, les garder en synchronisation…

clearskies ou syncthing ?

Ces deux projets semblent être les alternatives libre les plus connues à Bittorrent Sync; l'une est en Go et introduit une notion inutile (à mon avis) de nœuds en plus d'un *nouveau* protocole, qui fait (encore à mon avis) l'erreur d'être trop orienté fichier, au lieu d'avoir une vision globale du dossier, comme le fait git avec succès, et l'autre est en C++, que je ne connais pas assez pour pouvoir y contribuer. Oh et il a également son propre protocole.

Les autres ?

Il doit exister encore des dizaines d'alternatives que je n'ai pas citées, mais aucune d'elles n'a su me convaincre dans la simplicité d'installation et d'utilisation. Peut-être fais-je une erreur, l'avenir nous le dira.

L'avenir

Comme dit précédemment, il y a un minimum de points à implémenter avant de pouvoir prétendre avoir un remplacement à Bittorrent Sync, mais je compte justement garder les choses simples (le protocole, basé sur Bittorrent, a peu de chances d'évoluer même si c'est techniquement possible) pour que la communauté de développeurs puisse venir renforcer la bête. Dans un avenir lointain, il sera certainement possible de carrément remplacer Dropbox ou encore réinventer le web… mais ne brusquons pas les choses !

En attendant, vous pouvez jouer avec; n'hésitez pas à remonter les problèmes sur le tracker et à venir discuter sur la liste de diffusion (rakoshare servi-par librelist.com).

Note: ce document est placé sous licence CC0

(16 commentaires). Markdown Epub

 # Bravo

Posté par **JoeltheLion** (**page perso**) le 21/06/14 à 14:33. Évalué à 9 (+7/-0) .

> Bravo, il y a vraiment quelque chose à faire dans ce secteur. Je pense que tu as très bien cerné les besoins et les enjeux.

> Petite question: est-il prévu à un moment ou un autre de garder quelques versions des fichiers, comme Dropbox?

Répondre

[^] # Re: Bravo
Posté par **rakoo** (**page perso**) le 21/06/14 à 15:22. Évalué à 4 (+2/-0) .

> Merci pour les encouragements !
>
> Ce n'est pas prévu pour le moment, mais tout est fait pour que ce soit simple: en pratique chaque version du dossier a un seul et unique identifiant, le *hash* du torrent qui le décrit. On pourrait imaginer stocker ces versions et leur contenu dans un dossier spécial, éventuellement compressé, et fournir une GUI qui permette de naviguer dedans… Il y a beaucoup d'idées à reprendre de git et bup dans ce domaine, mais ça n'est pas encore la priorité :)

Répondre

[^] # Re: Bravo
Posté par **JoeltheLion** (**page perso**) le 21/06/14 à 15:36. Évalué à 3 (+1/-0) .

> Je pense que les projets de synchro qui se basent sur git utilisent une enclume pour écraser une mouche: git est prévu pour des logiciels, mais pour un logiciel de synchro seules quelques versions sont nécessaires, et tout le reste (branches, tags, etc.) est complètement superflu. Et d'ailleurs git n'est pas du tout fait pour stocker des binaires.
>
> Tu utilises un torrent par dossier? Est-ce que ça va marcher sur une arborescence avec des milliers de dossiers?

Répondre

[^] # Re: Bravo
Posté par **rakoo** (**page perso**) le 21/06/14 à 16:05. Évalué à 5 (+3/-0) .

> > tout le reste (branches, tags, etc.) est complètement superflu
>
> J'ai dit reprendre les idées, pas réutiliser :)
>
> Notamment, reprendre le modèle de donnés, qui permet de stocker des *blobs* de n'importe quel type (comme par exemple un torrent) et de les ordonner dans le temps. Tout le reste est du bricolage rendu possible parce que les fondations sont solides.
>
> > Tu utilises un torrent par dossier? Est-ce que ça va marcher sur une arborescence avec des milliers de dossiers?
>
> Un torrent par dossier *racine* partagé; les sous-dossiers ne font pas pas partie d'un autre torrent (à moins que l'utilisateur ait explicitement choisi de le partager… ce qui n'est même pas encore possible). La hiérarchie exacte n'a à peu près aucune influence sur le partage ("à peu près" parce que si c'est plus profond, le dictionnaire qui décrit le torrent sera plus gros, mais c'est le seul impact)

Répondre

 # Merci
Posté par **tuxicoman** (**page perso**) le 21/06/14 à 15:37. Évalué à 4 (+4/-1) .

> Je trouve le principe de Bittorrent Sync vraiment génial et c'est dommage que son code ne soit pas libre. Ca aurait fortement accéléré son adoption mais ca peut se comprendre si le business model de Bittorrent Inc en dépend.
>
> Mes petites questions bêtes :
> - Est ce que ca fonctionne si les 2 ordis sont derrière un NAT?
> - Est ce qu'un chiffrement des données transitées (données, ID et clé d'accès au dossier partagée) est prévue à terme?
> - Comment penses tu faire pour rendre cela multiplateforme ?

Répondre

[^] # Re: Merci
Posté par **rakoo** (**page perso**) le 21/06/14 à 16:34. Évalué à 10 (+12/-0) .

> Tout pareil: le concept de Bittorrent Sync est génial, dommage qu'il ne soit pas libre.
>
> > Est ce que ca fonctionne si les 2 ordis sont derrière un NAT?
>
> Pas testé, mais si le NAT-PMP marche "ça devrait".
>
> La solution que je vois est d'avoir une troisième machine qui fait tourner rakoshare et est accessible de l'extérieur. Techniquement cette troisième machine est un noeud tout à fait standard, mais en pratique elle va agir comme passerelle. Le but est qu'il n'y ait aucune configuration supplémentaire à faire, et qu'on ne s'embête pas avec ces bêtises de NAT.
>
> > Est ce qu'un chiffrement des données transitées (données, ID et clé d'accès au dossier partagée) est prévue à terme?
>
> Il n'y aura pas de clé au sens où tu l'entends; je compte réutiliser ce qui a été fait pour tahoe-lafs, à savoir avoir une suite de caractères qui contient toutes les informations nécessaires (autorisations, clé de chiffrement/déchiffrement, …) et qui ne doit avoir aucune signification pour

> l'utilisateur: ça doit être totalement opaque.
>
> Le chiffrement est prévu à 2 niveaux:
>
> - Les échanges devront bien entendu être chiffrés. Les récentes nouvelles sur OpenSSL m'ont un peu refroidi sur l'utilisation de TLS, d'autant plus qu'il n'y aura pas besoin de toute sa complexité. Je pense utiliser spiped qui a l'avantage d'être compréhensible dans son intégralité sans y passer des années
> - Les données elle-même devront être chiffrées indépendamment de l'échange. L'intérêt est de pouvoir envoyer les données à une machine tierce qui n'a pas les éléments nécessaires pour déchiffrer; cette machine tierce pourra agir comme passerelle (voir juste au-dessus) ou comme stockage longue durée (NAS, Dropbox, …). Bien sûr, encore une fois, rien de spécial à faire pour l'utilisateur si ce n'est *choisir* le bon id qui donne les bonnes capacités.
>
> > Comment penses tu faire pour rendre cela multiplateforme ?
>
> Par 2 moyens:
>
> - rakoshare est écrit en Go, et toutes les plateformes où rakoshare compile peuvent le faire tourner. Aujourd'hui, ça inclut DragonFly BSD, FreeBSD, Linux, NetBSD, OpenBSD, OS X (Darwin), Plan 9, Solaris et Windows; de quoi voir venir.
> - Plus important encore, je compte garder le protocole simple et le documenter, pour que des implémentations alternatives puissent voir le jour.
>
> Répondre

 # Syncthing
Posté par **rewind** le 21/06/14 à 16:17. Évalué à 8 (+6/-0) .

> l'une est en Go et introduit une notion inutile (à mon avis) de nœuds en plus d'un nouveau protocole, qui fait (encore à mon avis) l'erreur d'être trop orienté fichier, au lieu d'avoir une vision globale du dossier, comme le fait git avec succès

J'ai un autre avis. Je partage avec l'auteur de Syncthing l'idée que le protocole BT n'est pas fait pour faire de la synchronisation de dossier. Il y a plein de mécanismes existants dans le protocole BT qui sont complètement inutiles pour synchroniser des dossiers, notamment des mécanismes qui forcent un minimum au partage alors que dans le cas de la synchronisation, on veut partager donc pas besoin de forcer (je n'arrive plus à remettre la main sur le lien qui en parlait).

Et sur l'aspect fichier plutôt que dossier, je dirais qu'on verra à l'usage. Une synchronisation de dossier, ce n'est pas une gestion de version, même s'il y a une intersection non-vide entre les deux. Le protocole de Syncthing a l'avantage de la simplicité.

Répondre

[^] # Re: Syncthing
Posté par **rakoo** (**page perso**) le 21/06/14 à 16:48. Évalué à 10 (+8/-0) .

> Tu veux certainement parler du *tit-for-tat*, qui est effectivement inutile dans notre cas: ça tombe bien, je ne l'implémente pas !
>
> Mon calcul était plutôt de dire qu'il est plus facile d'adapter les innombrables bibliothèques bittorrent plutôt qu'inventer un protocole *from scratch*, et ainsi pouvoir réutiliser les infrastructures et surtout le travail existants.

Répondre

[^] # Re: Syncthing
Posté par **rewind** le 22/06/14 à 00:19. Évalué à 3 (+1/-0) .

> Oui, tu as raison, BT est probablement ce qui répond le mieux au problème. Maintenant, ce n'est pas forcément l'unique réponse et je trouve qu'explorer des alternatives est une bonne chose. Parce que si on voulait réutiliser l'existant, on aurait pu prendre FTP aussi. De toute façon, je préfère voir deux ou trois ou même quinze propositions de protocoles de synchronisation distribuée, plutôt que zéro comme c'était le cas jusqu'à il y a peu.

Répondre

 # Autre exemple "d'installation"
Posté par **tankey** le 21/06/14 à 16:22. Évalué à 8 (+6/-0) . Dernière modification : le 21/06/14 à 16:27

```
# yum install mercurial go
$ export GOPATH=/opt/go
$ go get github.com/rakoo/rakoshare
$ /opt/go/bin/rakoshare -fileDir ~/<mydir> -id <Id> -useLPD=true -useDHT=true
```

Répondre

Ultracopier
Posté par **alpha_one_x86** (**page perso**) le 21/06/14 à 23:07. Évalué à 1 (+4/-4) .

> Il y as aussi Ultracopier avec le plugin rsync, qui apporte simplicité un une GUI.
>
> –
>
> Mon projet libre: http://ultracopier-fr.first-world.info/, mon jeu libre: http://catchchallenger.first-world.info/

Répondre

ID
Posté par **flagos** (**page perso**) le 22/06/14 à 08:40. Évalué à 2 (+0/-0) .

> C'est quoi cette histoire d'ID ? C'est ce qui identifie ce qui est partagé ? Il n'y a aucune autre sécurité, n'importe qui qui récupère l'ID peut recuperer les données ?

Répondre

[^] # **Re: ID**
Posté par **Nils Ratusznik** (**page perso**) le 22/06/14 à 10:08. Évalué à 2 (+0/-0) .

> Comme l'auteur l'indique dans sa description, le logiciel ne semble pas totalement fini (il fonctionne, et c'est déjà pas mal ;)), et j'ose espérer qu'il prendra en compte ce genre de problématique pour la première version stable.

Répondre

[^] # **Re: ID**
Posté par **Xavier Claude** (**page perso**) le 22/06/14 à 10:23. Évalué à 2 (+1/-2) .

> Je ne vois pas la différence entre un ID à donner et un couple login/password. À un moment, il faut bien les renseigner pour que le logiciel puisse télécharger le bon fichier.
>
> –
>
> « Moi, lorsque je n'ai rien à dire, je veux qu'on le sache. » Raymond Devos

Répondre

[^] # **Re: ID**
Posté par **KiKouN** le 22/06/14 à 10:54. Évalué à 5 (+3/-0) .

> Le mieux serai une clé voir des clés asymétriques en plus de l'ID.
>
> Si tu n'as pas la clé, tu ne pourra pas chiffrer ou déchiffrer. Avec des clés asymétriques, on peut aussi donner des droits sur certains fichiers ou n'autoriser que certains transferts.
>
> Par exemple, un mobile ne pourra synchroniser uniquement vers un seul pair, limitant ainsi son traffic. Ce pair retransmettant alors aux autres pairs. Si le premier pair, n'est pas disponible, il suffit d'activer une autre clé public sur le mobile en attendant.
>
> Si le mobile est perdu, on peut invalider sa clé public sur les autres pairs.

Répondre

[^] # **Re: ID**
Posté par **rakoo** (**page perso**) le 22/06/14 à 15:07. Évalué à 8 (+6/-0) .

> Je suis désolé, la terminologie est un peu confuse, les détails ne sont pas encore là, mais voilà comment je vois les choses:
>
> Le système est calqué sur les capabilities de Tahoe-LAFS: les utilisateurs ne se soucient que de ça. C'est une chaine de caractères, opaque pour l'utilisateur (ie il copie-colle sans se poser de questions), qui contient toutes les informations nécessaires; voici en gros ce qu'il y a pour Tahoe-LAFS:
>
> - Est-ce que c'est un fichier ou un dossier
> - Est-ce que c'est mutable ou pas
> - Est-ce que j'ai le droit de modifier
> - Où est-ce que c'est stocké
> - Quelle est la somme de contrôle, de manière à ce que je puisse vérifier si le contenu est le bon
>
> Du coup, pour qu'un utilisateur donne accès à une ressource à quelqu'un d'autre, il lui suffit de copier-coller cet id. C'est d'ailleurs pour ça qu'on l'appelle également *capability*: ça indique les autorisations de celui qui connait cet id.
>
> Là où ça devient intéressant, c'est que tu peux *dériver* les *capabilities*: à partir d'une *capability* qui a les droits d'écrire, tu peux dériver une *capability* qui n'a que le droit de lire. Du coup si tu ne donnes que cette dernière, tu seras le seul à pouvoir modifier le contenu.
>
> Encore une fois, ce qu'il faut retenir, c'est que l'utilisateur n'a pas à comprendre le contenu de cet id; il doit juste copier-coller le bon (celui qui donne le droit d'écrire, celui qui ne donne que le droit de lire, celui qui ne permet que de stocker la version chiffrée mais pas de lire le contenu en clair).
>
> Après, ça veut dire que si tu colles cet id dans un email en clair qui passe chez Google, oui, Google aura les mêmes droits. Mais ça sera la même chose avec un login/mot de passe, et comme personne parmi les personnes visées n'utilise (voire connait) PGP, la sécurité n'est pas amoindrie. Disons que le point faible est dans la transmission de cette information de base, et ça sort du cadre de rakoshare. Je pars du principe que les utilisateurs sont un minimum conscients des personnes qui ont accès à ce qu'ils copient-collent dans une fenêtre de discussion ou un

email.

Note: pour plus de détails sur comment marche Tahoe-LAFS, voir cette présentation. Tahoe-LAFS est vraiment bon, j'espère que des fournisseurs de stockage auront une offre compatible.

Répondre

Envoyer un commentaire | Suivre le flux des commentaires

Note : les commentaires appartiennent à ceux qui les ont postés. Nous n'en sommes pas responsables.

Revenir en haut de page

À propos de LinuxFr.org

- Mentions légales
- Faire un don
- Team LinuxFr.org
- Informations sur le site
- Aide / Foire aux questions
- Règles de modération
- Statistiques
- API pour les développeurs
- Code source du site
- Plan du site